Assessment Test Solution Overview

Introduction

Ahoy! I'm Jorge Antonio Ramírez Padilla, a Software Developer with expertise in React Front End development that recently applied to your company, HolidayPirates.

I have been tasked to complete an Assessment Test as part of my Recruitment Process, so this document should be useful to explain the approach I took for my solution to this Assessment Test, with the objective to answer possible questions that may arise about my decisions and the code I have delivered.

Of course, if there are any more questions not covered in this document, feel free to reach out to me so we can discuss further this solution. I'm happy to help.

The Task

I was tasked to create a website where one can load a list of 5 Hotels. After loading said list, one can individually load the Reviews for that specific Hotel.

Setup

To access this API, I've utilized an environment variables file (.env) because it is a safe way to use keys to be later used in the app. In this case, to be able to make the app run, one has to follow the steps below:

1. Create a `.env` file.
2. Copy the contents of the `.env.dist` file located in the root of the project directory.
3. Get the SpaceID value and put it into `REACT_APP_CONTENTFUL_SPACE_ID`.
4. Get the Access Token value and put it into `REACT_APP_CONTENTFUL_ACCESS_TOKEN`.

Only for this project, I'll give you the SpaceID and Access Token you provided me in the document. Usually, one must share these types of access keys in a more secure way, but for the purpose of this project, I will provide them to you here for practical purposes:

```
SpaceID: gyfunrv4a4ak
Access Token: k9P9FQJcUpHKrHX3tXrgXunRyiS3qPchtY7V61tNruE
```

To run the program, simply run the following command:

```
$ npm start
```

The Process

In this section, I will discuss the technical choices I made for this project.

React TypeScript

I chose to use React instead of pure HTML/CSS and JavaScript, because using React I can showcase my area of expertise, while fulfilling the requirement of using those technologies with the use of TSX (TypeScript XML).

GraphQL

Given the Contentful API provided for data retrieval, I chose GraphQL due to its simplicity and efficiency in fetching data, and advantages such as unified endpoint access, efficient data fetching via single requests, and streamlined query composition, accelerating the development process.

Development

After understanding the project requirements, I divided tasks and reviewed designs. I then identified essential data elements required for display. I identified the following elements:

```
Hotels

- Hotel
  -- Image
  -- Title
  -- Rating
  -- Description
  -- Price
  -- Date Range

Reviews

- Review
  -- Feedback
  -- Username
  -- Comment
```

I took a look at the API, using the Contentful link provided to see how the API structure is. There, I realized that `hotelCollection` had a structure where I could get a list of hotels, each containing information just as I needed it. Even though the API already provides 5 hotels, I still decided to add the limit to the query, just in case the API has more Hotels in the future. The API provides extra information, which will be very useful if this project would be developed further as it is, but for this particular project I decided to create requests to only adhere to my needs.

I also realized that the `hotelCollection` provided a `imageCollection` of many images, linked to that particular hotel. But since for this project i only need to display one, I limited the query to only provide me one image.

Then, for requesting the Reviews for a Hotel, I used the `sys id` from the selected hotel to be able to request for the particular set of Reviews from the `reviewCollection`, where the hotelID was matching, making a lighter request each initial time. Then, I store these set of requests into a State Hook, so that it won't request the data again after it is once requested.

Testing

I integrated basic tests using React Testing Library to ensure code reliability. While these tests serve as a foundation, they can be expanded upon for comprehensive coverage. End-to-End Cypress tests could further improve testing capabilities.

Next Steps

I believe if one were to develop this particular app, there can be many improvements to be implemented. To name a few:

- Since the API gives us many images per Hotel, one could either create a gallery with buttons on the left and right, so the user could check from this view other pictures of that destination.
- We could also make it an automatic gallery, so that every couple of seconds, the image changes. For this, a smooth, horizontal scroll animation would work well.
- Hyperlinks: One could decide where to attach hyperlink in each hotel. For instance, it could be in the Hotel title to take us to the page of that Hotel.
- Hotel Ranking: I had this idea of being able to hover the Hotel Rating, and while hovering, getting an overview of how many times people have rated this hotel, how many are of each star (e.g. 3 of 1 star, 10 of 4 stars, 7 of 5 stars).
- A Router, if more pages were to be added to the project.

Conclusion

In conclusion, I must say this was a very satisfying and rewarding project to build. It allows me to show a bit of what I can do as a Front End Software developer as well as giving me a better understanding of how it would look like to be working with you and the types of content I would be seeing daily. I'm happy to discuss further my code, and any other questions about me, my work, or really anything else you might want to know.

Fair winds and may the tides be ever in your favor!